INTERACCIÓN CON EL ENTORNO Y COORDINACIÓN DE AGENTES FÍSICOS PARA LA REALIZACIÓN DEL *"JUEGO DE LAS SILLAS"* MEDIANTE ROBOTS AIBO

AUTORES: JOAN MIQUEL FUSTER MOLLÁ

FRANCISCO JOSÉ SÁNCHEZ MUNTÓ

TUTOR: MIGUEL ÁNGEL CAZORLA QUEVEDO

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL. UNIVERSIDAD DE ALICANTE

CURSO 2004/05

ÍNDICE

1. Introducción 3
1.1 Objetivo 3
1.2 Especificaciones robot AIBO y el lenguaje OPEN-R 3

2. Contenidos 5
2.1 Entorno de trabajo 5
2.2 Implementación 6

3. Resultados 9
3.1 Pruebas realizadas 9
3.2 Problemas encontrados 10

4. Conclusiones 11
5. Bibliografía 12

1. INTRODUCCIÓN

1.1 Objetivo

El objetivo de nuestro proyecto ha sido simular el juego de las sillas mediante robots AIBO.

Para ello hemos tenido que realizar un amplio estudio del lenguaje de programación Open-R, nos hemos ayudado de ejemplos realizados por SONY y las correspondientes modificaciones de éstos.

Para la realización de este proyecto hemos hecho uso del proyecto TeamChaos. Éste, es un proyecto realizado por varias universidades, entre ellas la Universidad de Alicante implementado para que los robots AIBO jugaran al fútbol. Esto nos ha ayudado ya que los módulos que más nos afectaban, como son movimiento, visión y comunicación, ya estaban implementados en este sistema y sólo ha habido que modificarlos. Se han añadido módulos propios como es el del sonido (módulo Sound), que posteriormente se pasará a explicar.

Para calibrar la cámara de los robots se ha usado la aplicación Chaos Manager, la cual usa el protocolo UDP para la comunicación con el robot.

1.2 Especificaciones robot AIBO y el lenguaje OPEN-R

AIBO es un robot de cuatro piernas que disfruta de un total de 20 grados de libertad, sus características más importantes para el modelo ERS-7 (modelo usado para realizar este proyecto) son las siguientes:

- CPU 64-bit RISC Processor 576 MHz (64 MB RAM)
- Program media Dedicated AIBO robot "Memory Stick™" media
- Head - 3 degrees of freedom; Mouth - 1 degree of freedom; Legs - 3 degrees of freedom x 4; Ears - 1 degree of freedom x 2; Tail - 2 degrees of freedom
- Cámara para visión
- Micrófonos en las orejas y altavoz
- Control de los motores de las piernas y cabeza
- Sensores de contacto en la cabeza y espalda
- Sensor de infrarrojos en el pecho
- Luces para expresar emociones
- Tarjeta Wireless
- Sistema operativo Aperios (Apertos) propietario Sony

Figura 1: robot AIBO ERS-7

El lenguaje por el cual el robot ha sido programado es Open-R. Open-R es la API que Sony está promoviendo actualmente para programar comportamientos propios. El OPEN-R SDK es un entorno de desarrollo basado en GCC (C++) mediante el cual se realiza el software que funciona en robots AIBO (modelos: ERS-7, ERS-210, ERS-220, ERS-210A, y ERS-220A).

En Open-R la unidad mínima de ejecución es un objeto Open-R. Un objeto Open-R es similar a un proceso UNIX. Cada fichero ejecutable en el programa final (con una extensión .bin) corresponde a un objeto. Cada objeto se comunica con otros objetos (se intercambian información) mediante el paso de mensajes.

Los objetos son *singlethreaded*. Esto significa que un objeto puede procesar un mensaje a la vez. Si un objeto recibe un mensaje mientras está procesando otro mensaje, el segundo mensaje se introduce en la cola de mensajes y será procesado más tarde.

Cuando dos objetos se comunican, al objeto que envía un mensaje se llama sujeto (subject) y al objeto que recibe el mensaje se le denomina observador (observer).

Para que un subject pueda mandar datos a uno o varios observers, al menos uno de ellos debe notificar estar preparado para recibirlos. Esto lo hace mandándole un “ReadyEvent" por medio de un “ASSERT_READY" al subject y su función es indicarle al subject que está preparado para recibir los datos. En ese momento el subject manda un “NotifyEvent" con los datos a los observers (si hubiera más de uno). Si en algún momento el observer no desea recibir datos, manda un “DEASSERT_READY" al subject indicándole que no está preparado.

2. CONTENIDOS

2.1 Entorno de trabajo

En cuanto al lugar de trabajo podemos diferenciar varias partes tal y como se muestra en la imagen:

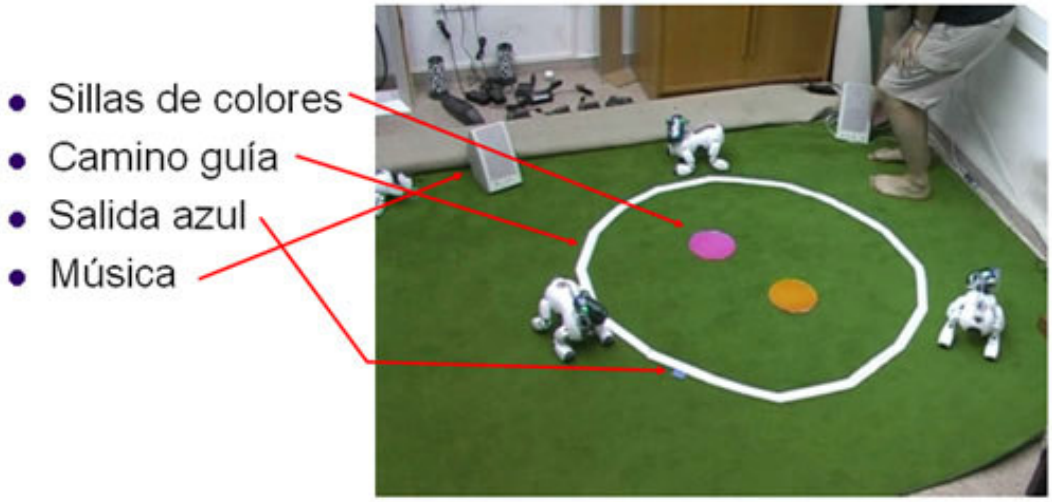

Figura 2: Entorno de trabajo

- **Sillas de colores**

Para simular las sillas del juego usamos círculos de colores de cartón en la moqueta.

- **Camino Guía**

Es el camino el cual guía a los robots por el terreno, los robot localizan la línea blanca e intentan mantenerla en el centro de la cámara por lo que van corrigiéndose si se desvían del camino.

- **Salida azul**

Para señalizarle la salida a los robots colocamos un cuadrado azul, cuando los robots pierden seguirán la línea blanca hasta encontrar la salida, con esto conseguimos que todos los perdedores vayan al mismo sitio.

2.2 Implementación

La ejecución está orientada a eventos, que son mandados al objeto y manejados por métodos públicos que definimos en cada objeto.

En nuestro caso, los eventos son:

- Sonido (ObserverSound): en este caso, el robot está continuamente recibiendo valores de sonido, para, en cualquier momento de la ejecución, saber si hay música o no.

- Mensaje Externo (ResultTcmExternalMessage): en este caso, el robot sólo envía un mensaje a otros robots cuando, durante la ejecución, se requiere del envío. Y todos los robots están continuamente esperando mensajes.

Para entender esto mejor, vamos a pasar a ver las distintas estructuras de datos y variables principales utilizadas en nuestra implementación.

Se ha creado un archivo de configuración inicial "*sound.ini*" en el que se incluyen las siguientes variables:

- numColours: número de colores actual.

- numRobots: número de robots que van a jugar en ese momento.

- Umbrales de Sonido (soundMax, soundMin): estos valores son los que deciden si hay o no hay sonido.

Variables globales más importantes usadas en el código principal:

- Boolean sound (true/false): usada para saber en cualquier momento del código si hay o no sonido.

- Boolean [colours] MaskColours: máscara de colores usada para saber qué colores tiene activos el robot en cada momento.

El formato de datos que recoge el AIBO por el micrófono y por tanto utilizado para el sonido es el denominado Pulse Code Modulated (PCM, modulación en código de pulso) de 8 bits. De esta forma se obtiene un byte por muestra de sonido.

Cuando al robot le llega un evento de sonido, exactamente le llega un byte de datos (*byte* data*), con el cual obtenemos el valor de Energía (W) de la siguiente forma:

$$energia = \sum_{i=0}^{1024} data(i)^2 \qquad (1)$$

$$energia = \frac{energia}{1024} \qquad (2)$$

Ahora tenemos el valor de la energía. Si este valor está en el rango entre *soundMAX* y *soundMIN* no pasará nada, si baja de *soundMIN* es que escucha sonido y si pasa de *soundMAX* es que no escucha sonido.

Debido a que los robots pueden estar en distintos estados, hemos implementado un autómata que lo refleja:

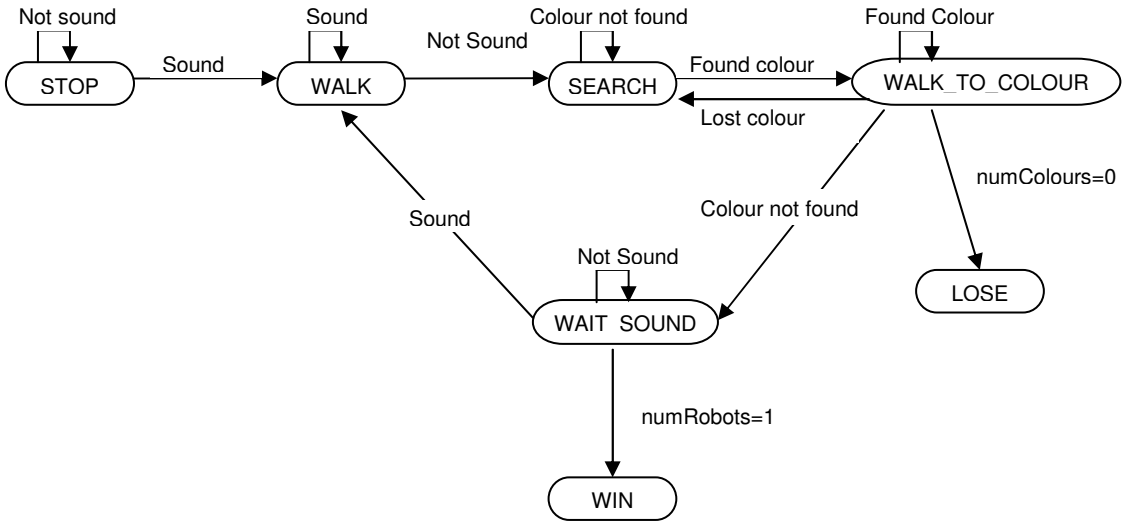

Figura 3: Diagrama de estados

- **Estado STOP**

Al inicio del juego todos los robots se encuentran en este estado, estarán alrededor del camino esperando a que empiecen a escuchar música, en ese momento cambiarán de estado y se pondrán a caminar siguiendo la línea blanca.

- **Estado WALK**

En este estado los robots seguirán la línea blanca mientras estén escuchando música. En cuanto paremos la música, el robot detectará que no hay música y cambiará el estado debido a que tiene que ir a buscar una “silla”, pasará al estado SEARCH.

- **Estado SEARCH**

El robot ya no está escuchando música, en este momento empieza a rotar hasta que visualiza alguna de las sillas, en este instante ya ha finalizado la búsqueda de sillas y cambia de estado, a WALK_TO_COLOUR.

- **Estado WALK_TO_COLOUR**

En este estado el robot ya ha elegido un color de silla y andará hacia este con el objetivo de apropiarse de la silla, así en este estado podemos encontrarnos ante dos situaciones:

- el robot llega a la silla, si se da este caso el robot parará en espera de que los demás hagan lo mismo y mandará un mensaje al resto comunicándole que la silla de su color ya esta ocupada, pasará al estado WAIT_SOUND.

- el robot recibe un mensaje, si no está buscando el color que le llega no pasa nada seguirá intentando llegar a su color, si le llega con el color que estaba buscando o volverá al estado SEARCH en busca de otro color o si no existen mas colores para buscar significará que se ha quedado sin silla por lo que habrá perdido (saldrá a la línea y buscará la salida) e ira al estado LOSE.

- **Estado WAIT_SOUND**

Los robots se quedarán esperando a que la música vuelva a ponerse en marcha, en ese momento volverá al estado WALK y empezará de nuevo todo pero con un color menos y un robot menos.

En el caso de que sólo haya un robot significará que será el campeón por lo que el juego terminará y pasará al estado WIN.

- **Estado LOSE**

Estado de finalización para un robot en concreto, cuando un robot llega a este estado significa que se ha quedado sin silla por lo que buscará el camino y lo seguirá hasta que encuentre la salida (marca azul) en ese momento saldrá del camino y finalizará su ejecución.

- **Estado WIN**

Estado de finalización, cuando se llega a este estado significa que el robot activo es el ganador y el juego terminará.

3.- RESULTADOS

3.1 Pruebas realizadas

Para la ejecución del juego debemos saber algunos conceptos previos:

Debido a que usamos los módulos del TeamChaos este tiene por defecto asignar números a colores, por ello para la realización del juego usamos los mismos, debido a esto debemos saber que los colores están organizados por el siguiente orden:

Colores: naranja (0), rosa (4), amarillo (1)...

Por tanto, para el caso de tres robots tendríamos dos colores. En la máscara de colores tendremos en la posición 0 y en la 4 un 1 que significa que el robot tiene activos esos colores y en las demás posiciones un 0.

En cada iteración quitamos un color en el orden anterior.

Realizamos distintas pruebas para tres robots con dos sillas y para cuatro robots con tres sillas, de esta manera pudimos apreciar como se complica el correcto funcionamiento simplemente añadiendo un robot más.

Figura 4: Ejemplo de ejecución con tres robots

Figura 5: Ejemplo de ejecución con cuatro robots

3.2 Problemas encontrados

Nos hemos encontrado con numerosos problemas los cuales vamos a separar en apartados:

- *Open-R*: insuficiente información sobre este lenguaje.

- *Sonido*: resulta complicado encontrar los umbrales de sonido óptimos (*soundMAX* y *soundMIN*) puesto que es complicado diferenciar entre sonido de ambiente y sonido musical.

- *Visión*: ha habido que hacer un continuo calibrado de la cámara puesto que a la mínima variación de iluminación los robots se descalibraban.

- *Comunicación*: al implementar la función de *Broadcast* nos hemos encontrado problemas como es que no llegue el mensaje adecuado a los demás robots o que al enviar el mensaje a los demás, estos se apagaban automáticamente.

- *Movimiento de las articulaciones*: al usar los métodos del TeamChaos y puesto que está orientado a que los robots jueguen al fútbol, en ocasiones se solapaban las conductas del TeamChaos con las nuestras.

4.- CONCLUSIONES

Como conclusión podemos decir que debido a como se ha realizado la implementación del código, siempre se ha buscado la generalidad, gracias a esto podemos ampliar el juego al número de robots que se quiera. Nuestro proyecto lo hemos centrado para tres y cuatro robots. Tan solo cambiando el archivo de configuración y añadiendo un numero superior de robots con el correspondiente aumento de las sillas (cartulinas de colores) y la ampliación del las medidas del camino (línea blanca) podemos hacer que funcione todo perfectamente.

Sin embargo la teoría no se corresponde con los resultados de la práctica, la ampliación del juego va seguida de numerosos problemas debido a que los robots no son perfectos y se producen numerosos errores por parte de los robots, el problema lo encontramos a que si uno de los robots tiene un fallo (error de visión, de sonido, etc...) repercute en la ejecución normal del juego ya que será imposible que el juego termine con éxito y provocará la cancelación.

Una mejora en la calidad de la cámara por parte de Sony haría que se produjeran menos errores, debido a que muchos de los problemas encontrados se centran en la visión (calibrado, iluminación, etc...).

5.- BIBLIOGRAFÍA.

- Web oficial de Sony.
http://www.aibo.com

- Tutorial y Manual Open-R Universidad de Murcia.
http://gsyc.escet.urjc.es

- Documentos y Ejemplos Open-R.
http://openr.aibo.com

- B. Bonev. (2005) Walk parameters optimization in a four-legged robot.